```python
import tkinter as tk
from tkinter import scrolledtext
from PIL import Image, ImageTk
import threading
import time
import cv2
import torch
import requests
import json
from datetime import datetime
import os

# Config
CITY = 'San Diego'
API_KEY = '5212b154f598c937b1ff3e98853169de'  # вЊ… Your actual API key
DETECTION_INTERVAL = 2
WEATHER_INTERVAL = 5

# Voice for macOS
def speak(text):
    print("Speaking:", text)
    os.system(f'say "{text}"')

# Get weather
def get_weather():
    try:
        url = f"http://api.openweathermap.org/data/2.5/weather?q={CITY}&appid={API_KEY}"
        data = requests.get(url).json()
        print("Weather API Response:", data)
        return data['weather'][0]['main']
    except Exception as e:
        print("Weather fetch failed:", e)
        return "Unknown"

# Load YOLOv5
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', trust_repo=True)

# Write JSON
def write_json(weather, detections):
    output = {
        "weather": weather,
        "timestamp": datetime.now().isoformat(),
        "objects": detections
    }
    with open("inference_output.json", "w") as f:
        json.dump(output, f, indent=2)

# Object detection
def run_object_detection():
    cap = cv2.VideoCapture(0)
    while True:
        ret, frame = cap.read()
        if not ret:
            continue

        results = model(frame)
        detections = []

        for *xyxy, conf, cls in results.xyxy[0]:
            label = model.names[int(cls)]
            detections.append({"class": label})

            x1, y1, x2, y2 = map(int, xyxy)
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.putText(frame, label, (x1, y1 - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

        write_json(current_weather[0], detections)
        cv2.imwrite("latest_frame.jpg", frame)
        time.sleep(DETECTION_INTERVAL)

    cap.release()

# Weather update
def update_weather_loop():
    while True:
        current_weather[0] = get_weather()
        print("Weather updated to:", current_weather[0])
        time.sleep(WEATHER_INTERVAL)

# Main UI App
class VoiceUIApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Voice Assistant UI")
        self.root.geometry("640x600")

        self.status_label = tk.Label(root, text="System Status: Running", bg="lightgreen")
        self.status_label.pack(fill=tk.X)

        self.text_feed = scrolledtext.ScrolledText(root, height=8)
        self.text_feed.pack(fill=tk.BOTH, padx=10, pady=5, expand=True)

        self.alert_feed = scrolledtext.ScrolledText(root, height=4, bg="#fdf5e6")
        self.alert_feed.pack(fill=tk.BOTH, padx=10, pady=5, expand=True)

        self.image_label = tk.Label(self.root)
        self.image_label.pack(pady=5)

        self.mute = False
        self.previous_alerts = set()  # вЊ… Alert history

        self.setup_buttons()
        self.poll_data()

    def setup_buttons(self):
        frame = tk.Frame(self.root)
        frame.pack(fill=tk.X, pady=5)
        self.mute_btn = tk.Button(frame, text="Mute", command=self.toggle_mute)
        self.mute_btn.pack(side=tk.LEFT, padx=10)
        tk.Button(frame, text="Clear Alerts", command=self.clear_alerts).pack(side=tk.RIGHT, padx=10)

    def toggle_mute(self):
        self.mute = not self.mute
        self.mute_btn.config(text="Unmute" if self.mute else "Mute")

    def clear_alerts(self):
        self.alert_feed.delete("1.0", tk.END)
        self.previous_alerts.clear()  # вЊ… Clear voice alert memory too

    def poll_data(self):
        try:
            with open("inference_output.json") as f:
```

```
                data = json.load(f)
            self.update_ui(data)
        except Exception as e:
            print("Failed to read inference_output.json:", e)

        try:
            image = Image.open("latest_frame.jpg")
            image = image.resize((320, 240))
            photo = ImageTk.PhotoImage(image)
            self.image_label.config(image=photo)
            self.image_label.image = photo
        except:
            pass

        self.root.after(3000, self.poll_data)

    def update_ui(self, data):
        self.text_feed.delete("1.0", tk.END)
        alerts = self.generate_alerts(data)

        self.text_feed.insert(tk.END, f"Weather: {data.get('weather')}\n")
        for obj in data.get("objects", []):
            self.text_feed.insert(tk.END, f"Detected: {obj['class']}\n")

        self.alert_feed.delete("1.0", tk.END)
        for alert in alerts:
            self.alert_feed.insert(tk.END, f"{alert}\n")

        if not self.mute and alerts:
            threading.Thread(target=self.speak_alerts, args=(alerts,), daemon=True).start()

        if data.get("weather") in ["Storm", "Fog"] and any(obj["class"].lower() == "person" for obj in data["objects"]):
            self.root.config(bg="red")
        elif data.get("weather") in ["Rain", "Fog"]:
            self.root.config(bg="yellow")
        else:
            self.root.config(bg="lightgreen")

    def generate_alerts(self, data):
        alerts = []
        weather = data.get("weather")
        if weather == "Fog":
            alert = "Severe fog detected. Turn on high beams."
            if alert not in self.previous_alerts:
                alerts.append(alert)
                self.previous_alerts.add(alert)

        for obj in data.get("objects", []):
            if obj["class"].lower() == "person":
                alert = "Pedestrian detected. Please slow down."
                if alert not in self.previous_alerts:
                    alerts.append(alert)
                    self.previous_alerts.add(alert)
            if obj["class"].lower() == "stop sign" and weather == "Fog":
                alert = "Fog and stop sign ahead. Prepare to stop."
                if alert not in self.previous_alerts:
                    alerts.append(alert)
                    self.previous_alerts.add(alert)

        return alerts

    def speak_alerts(self, alerts):
        for alert in alerts:
            speak(alert)
            time.sleep(1.5)

# Launch
current_weather = ["Clear"]

if __name__ == "__main__":
    threading.Thread(target=run_object_detection, daemon=True).start()
    threading.Thread(target=update_weather_loop, daemon=True).start()

    root = tk.Tk()
    app = VoiceUIApp(root)
    root.mainloop()
```